

Chapter-1 : Introduction

❖ Chapter - 1 Introduction:

10 Hrs

Definition of Software, Software Engineering, Software Products and Software process, Process models: Waterfall modal, Evolutionary Development, Bohemia's Spiral model, Overview of risk management, Process Visibility, Professional responsibility. Computer based System Engineering: Systems and their environment, System Procurement, System Engineering Process, System architecture modelling. Human Factors, System reliability Engineering. Requirements and Specification: The requirement Engineering Process, The Software requirement document, Validation of Evolution of requirements, Viewpoint – oriented & method based analysis, system contexts, Social 7 organizational factors. Data flow, Semantic, Objects, models, Requirement Specification, Non functional requirement.

Software:

It is a program or a set of programs that contains instructions to perform a specific task.

It also includes the rules and related data documents of the task to be done.

Types of software:

- a) System software

- a) **System software:** Kind of software designed to operate the computer hardware. It provides an interface for the users to perform their tasks.

Ex : Operating System, Language Translators, System utility software like linkers, loaders etc.,

- b) Application software:** Kind of software designed for the users to perform their task.

Ex : Office automation system, general purpose software like calculator, paint, WordPad etc.,
Special purpose software like payroll system, stock maintenance etc.,

Software Engineering :

“It is an approach to develop a software product with well-defined scientific principles, methods and procedures.”

The outcome of this approach is an efficient reliable software product.

[According to IEEE,

“The application of a systematic, disciplined and quantifiable approach to the development, operation, maintenance of software. “]

Characteristics of a Good Software :

- **Efficient software:** Should make efficient use of resources like processor, memory, I-O devices etc.,
- **Dependency:** A software should be very secured and safer that should not cause any damage in case of system failure.
- **Maintenance:** A software should maintainable in such way that it should meet the future enhancements if needed.
- **Usable software:** It must provide a user-friendly and easy interface to users along with necessary documentation to handle it.

Goals of Software Engineering:

- Improve the quality of software products.
- To increase productivity
- To get the ability to develop and maintain a software,
- Job satisfaction to software engineers.

Components of Software Engineering:

1. **SDLC (Software Development Life Cycle) :**
Includes different stages & activities of software development.
2. **SQA (Software Quality Assurance)**
Process of ensuring the user satisfaction through the quality product development.
3. **SPM (Software Project Management)**
Application of principles of project management to the software development process.
4. **SM (Software Management):**
Includes methods & procedures to be followed for effective maintenance of software & change controls.
5. **CASE (Computer Aided Software Engineering):**
They are automatic tool used to support the process of software development.

Software Product:

It is the software (final product) delivered to the customer with necessary documentation to use it properly.

Types of Software Products:**1. Generic products****2. Customized(Be Spoke) products****1. Generic Products:**

- They are general software products developed to sell any customer.
- They are designed mainly for general marketing.
- The developer controls the requirements & specification.

Ex : Spreadsheet software, Word Processor, Drawing software tools.

2. Customized Products:

- They are developed as per the customer requirement.
- Customer controls the requirements & specification.
- Analysis of user requirement needs to be done.

Ex : Hospital Management, Railway reservation

Software Process :

It is a set of activities used to produce a software product.
Different stages are applied to develop a reliable software product.

The basic activities involved in software process are**1. Software Specification:**

A detailed description of the software developed with functional and non-functional requirements are mentioned here

2. Software Development:

In this process, designing, programming, documenting, testing and bug fixing is done.

3. Software Validation:

In this process, we check whether evolution of software product is done to check whether the software meets end user's needs and business requirements.

4. Software Evolution:

It is the process of developing software initially, then updating it frequently for various reasons.

Software Process Model :

It is the simplified representation of a software process.

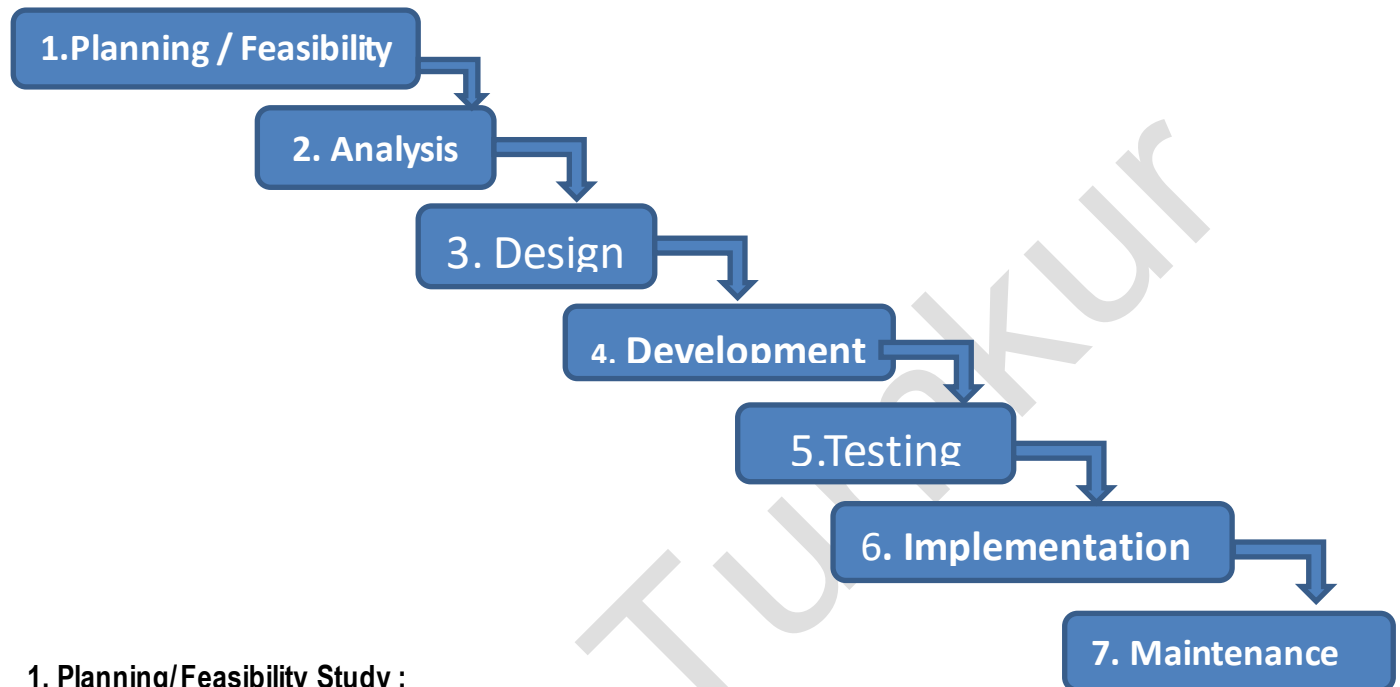
OR

It is an abstraction of the actual software process.

Each model represents a process from a specific perspective.

**Software Development Life Cycle:**

It refers to a methodology with clearly defined processes for creating high quality software. It is carried out in different stages. Each stage will use the result of previous stage as input.

**1. Planning/Feasibility Study :**

This stage helps to define the problems and scope of the existing systems and helps to define the objectives of new system.

2. Analysis stage:

- It includes gathering of all specific requirements for a new system.
- Developers will create a software requirement specification or SRS document.
- This includes all the specifications for hardware, software & network requirements for development

3. Design Stage:

In this stage the new system is designed as per the user's needs. Here, the solution problem is found for given problem and specification for each and every component such as user interface, system interface, network and databases

4. Development Stage:

This is the stage, where developers actually write code and develop the system according to the design documents.

5. Testing :

This is the stage, the developed software is tested whether it functions properly or not and also there are no errors. It is important that the software should meet the overall standards specified in SRS.

6. Implementation :

In this stage, all the modules are integrated with primary source code and installed at client's place.

7. Maintenance :

In this stage, developer meet the client's place to check whether the users have any issues and bugs which have to be solved.

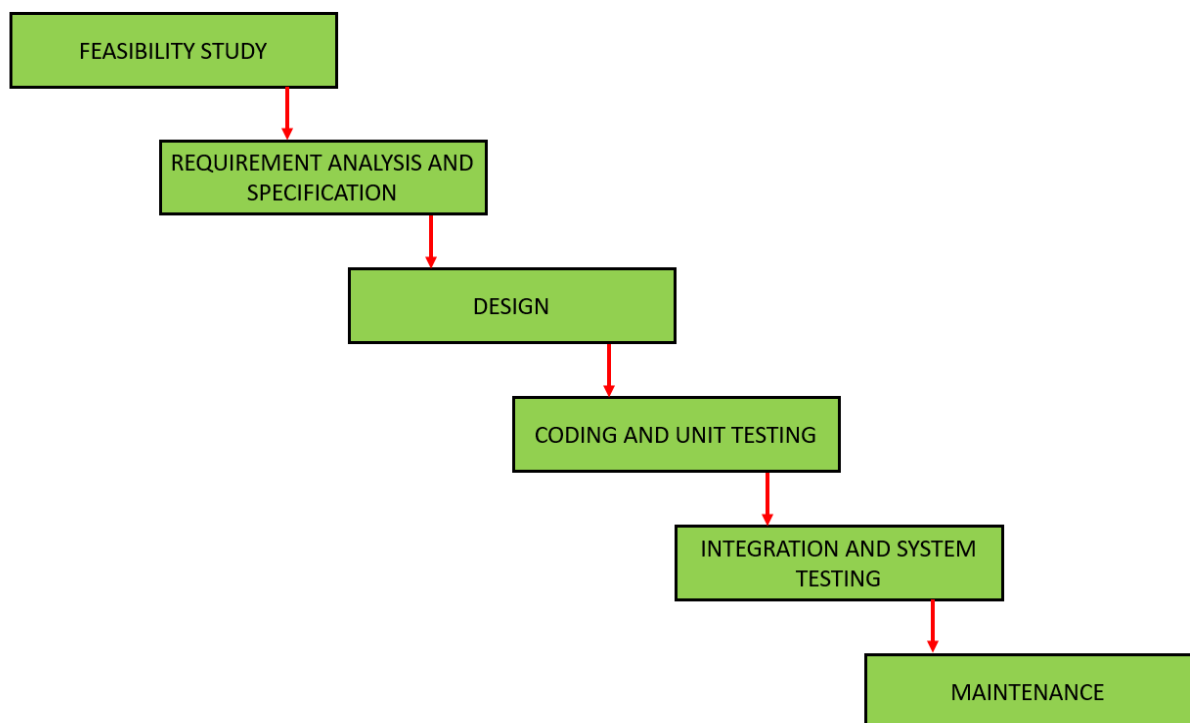
It includes adding improvements and updates the new versions.

Types of SDLC Models

- Water fall model
- Evolutionary Development model
- Bohemia's Spiral model

Water fall model

- It is a software life cycle described by W.W. Royce in 1970.
- The water model is also referred as linear sequential model or classical life cycle model.
- The development is supposed to proceed linearly through the phases.
- In a waterfall model, each phase must be completed fully before the next phase can begin.
- This type of model is basically used for the project which is small and there are no uncertain requirements.



1. Feasibility Study:

- The main goal of this phase is to determine whether it would be financially and technically feasible to develop the software.
- The feasibility study involves understanding the problem and then determines the various possible strategies to solve the problem.

2. Requirements analysis and specification :

- The aim of this phase is to understand the exact requirements of the customer and document them properly.
- It includes two activities: **requirement gathering, analysis and requirement specification.**



Requirements for the software development is gathered and analyzed to remove the incompleteness and inconsistencies.

The analyzed requirements are documented in SRS document which acts as an agreement between the customer and developer.

3. Design:

In this stage, the requirements specified in SRS document is transformed into a structure that can be implemented in a programming language.

4. Coding and Unit testing:

In this stage, the design structure is translated into program source code using programming language.

The aim of the unit testing phase is to check whether each module is working properly or not.

5. Integration and System testing:

Integration of different modules are undertaken soon after they have been coded and unit tested. Finally, after all the modules have been successfully integrated and tested, the full working system is obtained and system testing is carried out on this.

- a) **Alpha testing:** Alpha testing is the system testing performed by the development team.
- b) **Beta testing:** Beta testing is the system testing performed by a friendly set of customers.
- c) **Acceptance testing:** The actual customer will test the delivered software to determine whether to accept or reject it.

6. Maintenance:

Maintenance is the most important phase of a software life cycle. The effort spent on maintenance is the 60% of the total effort spent to develop full software.

Advantages of Water fall model

1. This model is very simple and is easy to understand.
2. Phases in this model are processed one at a time.
3. Each stage in the model is clearly defined.
4. This model has very clear and well understood milestones.
5. Process, actions and results are very well documented.
6. This model works well for smaller projects and projects where requirements are well understood

Disadvantages of Water fall model

1. In this model, the risk factor is higher, so this model is not suitable for more significant and complex projects.
2. This model cannot accept the changes in requirements during development.
3. It becomes tough to go back to the phase.
4. Since the testing done at a later stage, it does not allow identifying the challenges and risks in the earlier phase, so the risk reduction strategy is difficult to prepare

When to use Waterfall Model?

Some Circumstances where the use of the Waterfall model is most suited are:

- When the requirements are constant and not changed regularly.
- A project is short
- The situation is calm
- Where the tools and technology used is consistent and is not changing
- When resources are well prepared and are available to use.

Evolutionary Model

Evolutionary model is also referred to as the **successive versions model** and sometimes as the **incremental model**.

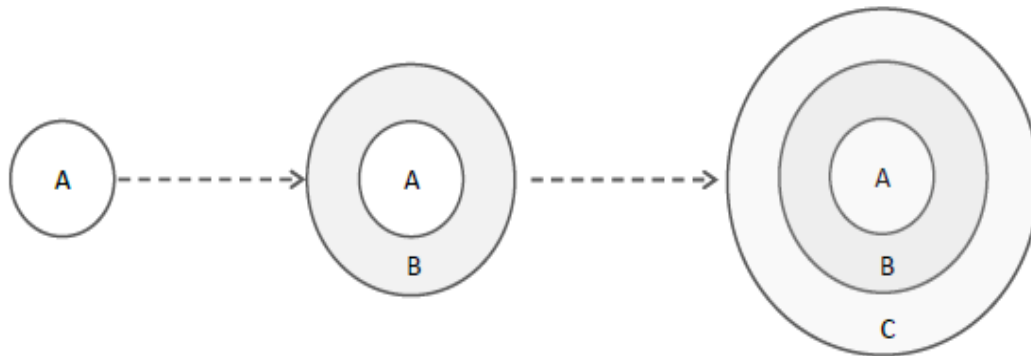
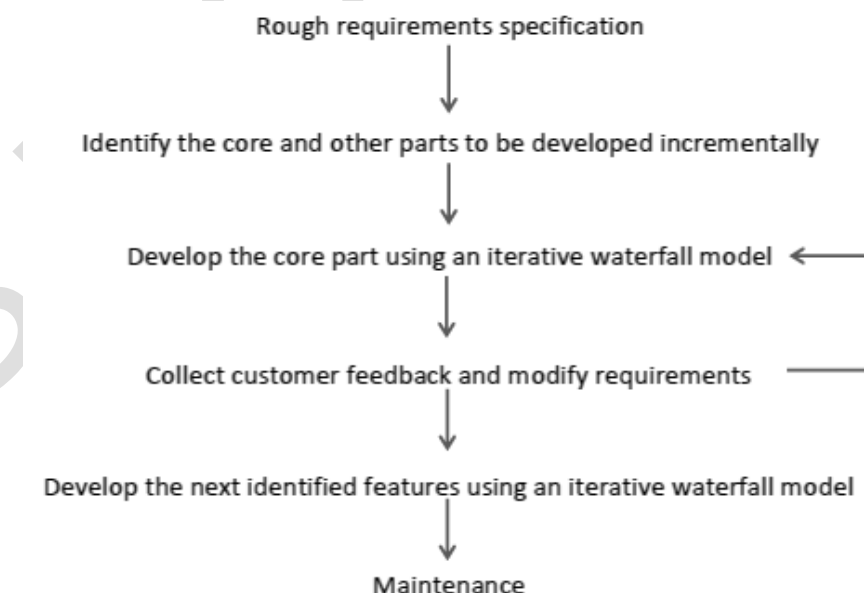


Figure 5: Evolutionary Development of a Software Product

- In Evolutionary model, the software requirement is first broken down into several modules (or functional units) that can be incrementally constructed and delivered.
- The development first develops the **core modules** of the system.
- The core modules are those that do not need to depend on other modules.
- The initial product structure is refined into increasing levels of capability by adding new functionalities in successive versions.
- Each evolutionary model may be developed using an iterative waterfall model of development.
- The evolutionary model, each successive version/model of the product is a fully functioning software capable of performing more work than the previous versions/model.
- The evolutionary model is normally useful for very large products, where it is easier to find modules for incremental implementation.



Often, **evolutionary model** is used when the customer prefers to receive the product in increments so that he can start using the different features as and when they are developed rather than waiting all the time for the full product to be developed and delivered.

Advantages of Evolutionary Model:

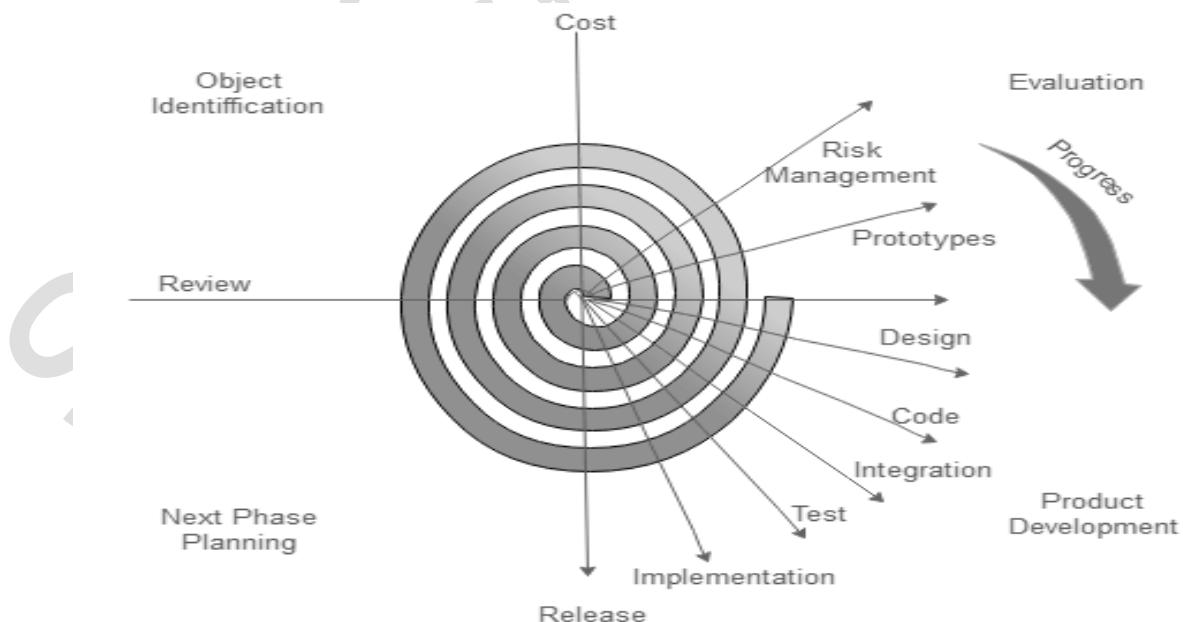
- **Large project:** Evolutionary model is normally useful for very large products.
- User gets a **chance to experiment with a partially developed software** much before the complete version of the system is released.
- Evolutionary model helps to accurately **bring out user requirements** during the delivery of different versions of the software.
- The core modules get tested thoroughly, thereby **reducing the chances of errors** in the core modules of the final products.
- Evolutionary model **avoids the need to commit large resources** in one go for development of the system.

Disadvantages of Evolutionary Model

Difficult to divide the problem into several versions that would be acceptable to the customer and which can be incrementally implemented and delivered.

Bohemia's Spiral Model

- **This model was proposed by Boehm, which later became Bohemia's (1988)**
- **Spiral model** is one of the most important Software Development Life Cycle models, which provides support for **Risk Handling**.
- This model contains many loops which are unknown and can vary from project to project based on project risk.
- Each loop represents by each loop.
- Project manager plays an important role as he decides the number of phases dynamically.
- The radius of spiral at any point represents the expenses of project so far.
- The angular dimension represents the progress made so far in the current phase.



Each cycle in the spiral is divided into four parts:**a) Objective setting(Object Identification):**

Objective of the phases is determined. Constraints on the process and products are identified. Detailed management plan is drawn up. Project risks are identified.

b) Risk Assessment and reduction(Evaluation):

For each identified project risks, a detailed analysis is carried out. Steps are taken to reduce the risk.

c) Development and validation(Product Development):

After evaluation, a development model for the system is chosen. The decision of choosing the development model is guided by the risk factors.

d) Planning(Next phase Planning):

It involves reviewing the results of the preceding stages. After that decision is made whether to continue or not with a further loop of the spiral. If it is decided to continue, plans are drawn up for the next phase of the project.

When to use Spiral Model?

- When deliverance is required to be frequent.
- When the project is large
- When requirements are unclear and complex
- When changes may require at any time
- Large and high budget projects

The advantages:

- Changing requirements can be accommodated.
- Allows extensive use of prototypes.
- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.

The disadvantages:

- Management is more complex.

- End of the project may not be known early.
- Not suitable for small or low risk projects and could be expensive for small projects.
- Process is complex
- Spiral may go on indefinitely.
- Large number of intermediate stages requires excessive documentation.

Overview of Risk Management

“Risk is anything that may affect the achievement of an organization’s objectives”.

“It is an uncertain event or condition that if it occurs has positive or negative effects on a project’s objectives”.

Risk management is the process of identifying, assessing, and prioritizing the risks to minimize, monitor, and control the probability of unfortunate events.

“A risk management is a document that a project manager prepares to foresee risks, estimate impacts and define response to it”.

A risk management includes “Risk assessment and Risk control”

Risk assessment is the “product of probability of risks that may occur and severity of the risks.”

Risk control is controlling the identified risks based on their priority.

Impact Risk assessment matrix:

Risk Assessment=probability of risk x severity of risk

		Severity of risk			
Probability of risk		Negligible-1	Marginal-2	Critical-3	Catastrophic-4
	Frequent-5	Medium-5	High-10	Very high-15	Very high-20
	Likely-4	Medium-4	High-8	High-12	Very high-16
	Occasional-3	Low-3	Medium-4	Medium-6	High-12
	Unlikely-2	Low-2	Medium-4	Medium-6	High-8
	Rare-1	Low-1	Low-2	Low-3	Medium-4

Two types of engineers can handle risk:

- **Reactive Engineer**: corrects the problem as it occurs.

Reactive risk management tries to reduce the damage of potential threat and speed on organization's recovery from them, but assumes that those threats will happen eventually.

- **Proactive Engineer:** Starts thinking about possible risks that might occur in a project before they occur.

Proactive risk management identify threats and aims to prevent those events from ever happening in the first place.

There are different types of risk can occur during software development

1. **Generic risk:** It occurs during requirement change, loss of team member and loss of funding.
2. **Project risk:** It affects the project schedule (duration).
3. **Product risk:** It affects quality and performance of the product.
4. **Business risk:** It affects the viability(ability to survive) of the product.

Ex: In 2005, Denver International Airport was set to create a most sophisticated luggage management system in world. The project was seem to be far more complex than anticipated and delayed by 16 months at a cost in \$560 million.

The 4 essential steps of the Risk Management Process are:

1. Risk Identification
2. Risk Analysis
3. Risk Planning
4. Risk Monitoring.

The process of risk management involves several stages are as follows-

1. Risk Identification:

In this stage, the possible project, product and business risks are identified.

2, Risk Analysis:

In this stage or process, the likelihood and consequences of these risks assessed.

3. Risk Planning:

In this stage, risk avoidance is either planned to affect the plan or reduce its effects on the project.

4. Risk Monitoring:

In this stage, risk assessment is done continuously and the risk reduction plan is revised as more information about risk is available.

Process visibility:

It means to ability to measure the progress of the software development process.

Software process managers need documents, reports and reviews to keep track of what is going on. Each activity or phase must end with production of same document.

These documents make software process visible. Visibility of some process models are :

Process Model	Rating	Process Visibiliy
---------------	--------	-------------------

Water fall model	Good Visibility	Each activity produces some deliverable
Evolutionary Model	Poor visibility	uneconomic to produce documents during rapid iteration
Spiral Model	Good visibility	each segment and each ring of the spiral should produce some document.

Professional & Ethical Responsibilities:

Software Engineers must behave honest & ethically responsible. Some of the responsibilities are,

- **Confidentiality :**

Engineers should have respect over their employees, clients and also maintain the secrets and also respect the confidentiality of agreements etc.

- **Competence:**

Engineers should not miss-represent their level of competency. They should not knowingly accept work which is out of their acceptance and capability.

- **Computer Misuse:**

Software engineers should not use their technical skills to misuse other computers and for other purpose.

- **Intellectual property Rights:**

Engineers must be aware of local laws governing the use of intellectual properties such as patents, Copyrights, etc.

System Procurement

It is the process of acquiring a system for an organization to meet predefined requirement and specification.

Procurer: The person who obtains or acquires the system.

Specification and architectural design of a system should be known before procurement in order to buy or let a contractor to design and build a system of high-level specification.

System Procurement Process

This is the *process* to identify sources that could provide needed products or services for the acquiring organization.

Types of Procurement System are:

- **Commercial Off The Shelf (COTS) System.**
- **Contractor and Sub Contractor Method.**

COTS(Commercial off-the-shelf)

Commercial-off-the-shelf (COTS) software is a term for software products that are ready-made and available for purchase in the commercial market.

Contractor and Sub Contractor Method.

- The Contractor or Sub contractor minimizes the task of the System Procurement.

- The Sub contractor builds and design parts of the System
- The different parts designed by subcontractor are integrated by the principal contractor then is delivered to the system customer.
- Depending upon the contract the principal contractor going to select the Number of Sub contractor.

System Procurement

It is the process of acquiring a system for an organization to meet predefined requirement and specification.

Procurer:

The person who obtains or acquires the system.

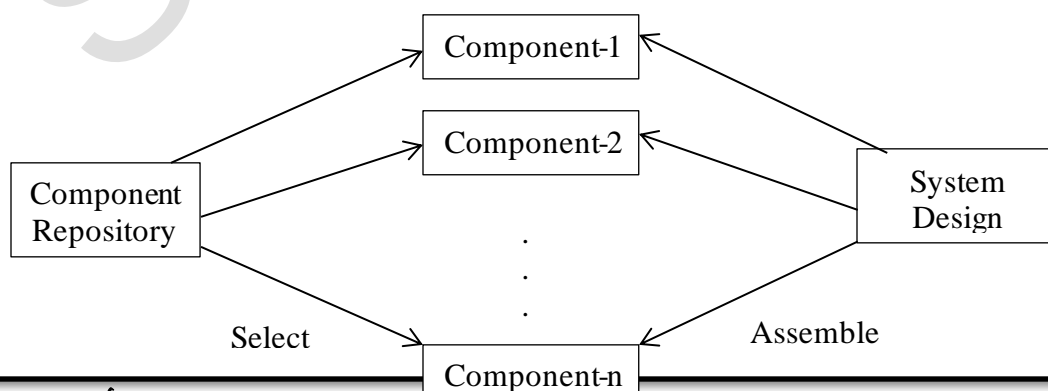
Specification and architectural design of a system should be known before procurement in order to buy or let a contractor to design and build a system of high level specification.

Types of Procurement System are:

- **Commercial Off The Shelf (COTS) System.**
- **Contractor and Sub Contractor Method.**

COTS(Commercial off-the-shelf)

Commercial-off-the-shelf (COTS) software is a term for software products that are ready-made and available for purchase in the commercial market.



It is a term used to describe the purchase of a product that are standard manufactured products and purchase the system which makes the specification using COTS. These products usually do not meet the requirements exactly, unless the requirements have been written with specification. Therefore choosing a system means finding the closest match b/n the system requirements and the facilities offered by COTS.

Contractor and Sub Contractor Method.

The Contractor or Sub contractor minimizes the task of the System Procurement. The Sub contractor builds and design parts of the System. The different parts designed by subcontractor are integrated by the principle contractor then is delivered to the system customer. Based on the contract, the principle contractor going to select the number of sub-contractor.

System Procurement Process

This is the *process* to identify sources that could provide needed products or services for the acquiring organization.

Computer Based System Engineering

System :

It is a collection of inter-related components that work together to achieve some objective.

System Engineering:

It is the activity of specifying, deciding, implementing, validating, installing and maintaining system as a whole.

“System Engineering is an interdisciplinary field of engineering that focuses on how to design and manage complex engineering systems over their life cycles to form a complete system”.

Emergent Properties:

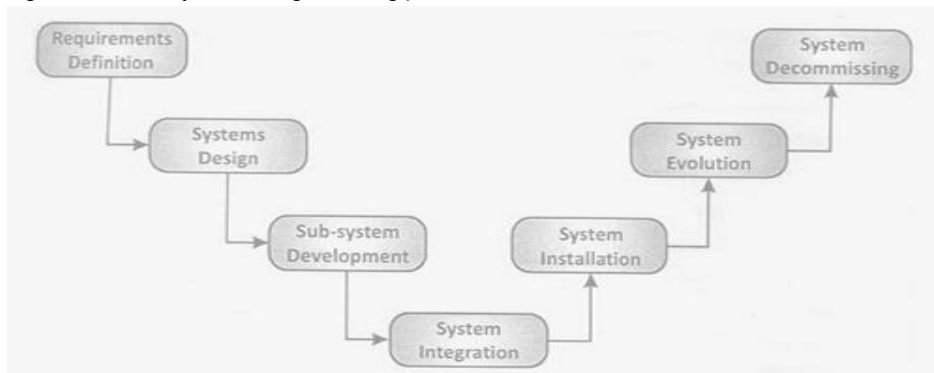
Properties of the system as a whole rather than the properties that can be derived from the components of a system.

- **The overall weight:** This can be computed from individual component property.
- **The reliability of the system:** This depends on the system components and the relationship between the components.
- **The usability of the system:** This depends on the system operation and the way in which it is used in the environment.

System Engineering Process:

System engineering process is the inter-disciplinary process that ensures that the customer needs are satisfied throughout system life cycle.

Below figure shows System engineering process,



The phases of System engineering process are,

1. Requirement definition
2. System design process
3. Sub System development
4. System integration
5. System installation
6. System evaluation
7. System decommissioning

1. System requirement (Requirement definition)

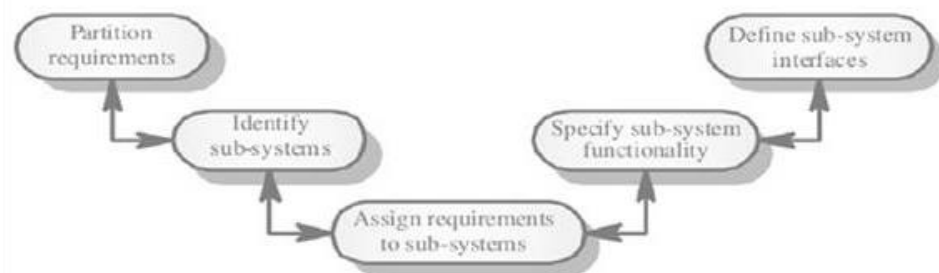
- It specifies what the system should do and its essential & desirable system properties.
- It also involves consultations with system customers & end users.

This phase concentrates mainly on three properties: -

- a. **Abstract functional requirements:** The basic functions that system must provide are defined at this level.
- b. **System properties:** These are Non-functional emergent system properties such as availability, performance & security, this affect the requirements for all subsystems.
- c. **Characteristic that system might not exhibit:** It is to specify what the system should not do as it specifies what the system should do.

2. System Design:

This is the main phase in System engineering; it is concerned with how the functionality of the system has to be provided by the components of the system. It mainly consists of five phases,



Five phases are:

- **Partition Requirements**

Analyzing the requirements and organize them into related groups.

- **Identifying subsystems**

To identify subsystems that can individually or collectively meet the requirements. Groups of requirements are usually related to subsystems and partitions will be done together.

- **Assign Requirement to subsystems**

Here the requirements are assigned to each identified subsystem. If there is no clean match for requirements to system, it will be asked to design or COTS systems should be modified.

- **Specify sub system functionality**

The specific functions provided by each sub system are specified in order to identify the relationship between the subsystems.

- **Define Sub System interfaces**

This is to define the interfaces that are provided and expected by each sub system. Once these interfaces have been agreed the parallel development of the sub system becomes possible.

3. Sub System Development:

The sub system development activity involves developing each of the sub system identified during system design. This may enter another system engineering process for the sub system.

4. System Integration:

System integration involves taking independently the developed sub system and putting them together in order to make up a complete system.

Integration can be done in two ways

- Big Bang Method** : All the sub systems are integrated at the same time.
- Incremental Method**: Sub systems are integrated one by one. It is a best approach because of two reasons,
 - It usually impossible to schedule the development of all sub system so that they can be finished at a time.
 - It reduces cost of error location.

5. System installation

It is the activity of installing the system in the environment where it is intended to operate.

Different problems that can occur during system installation are,

- Environment assumption may be incorrect.
- Operator training has required.
- There may be physical installation problems.
- May be human resist to introducing the new system.

6. System Evolution

Large & complex systems having very long life time, during their life time some system evolution takes place.

Evolutions are costly for a reason:

- Purpose changes have to be analyzed from a technical perspective.
- Sub system are never independent, changes made to one sub system may affect the performance of another sub system and the functionality of a whole.
- The system age & their structure become corrupted by change.

7. System Decommissioning

It means taking the system out of service after the end of its use full life time for example, hardware components of a system can be reassembled and reused in another system.

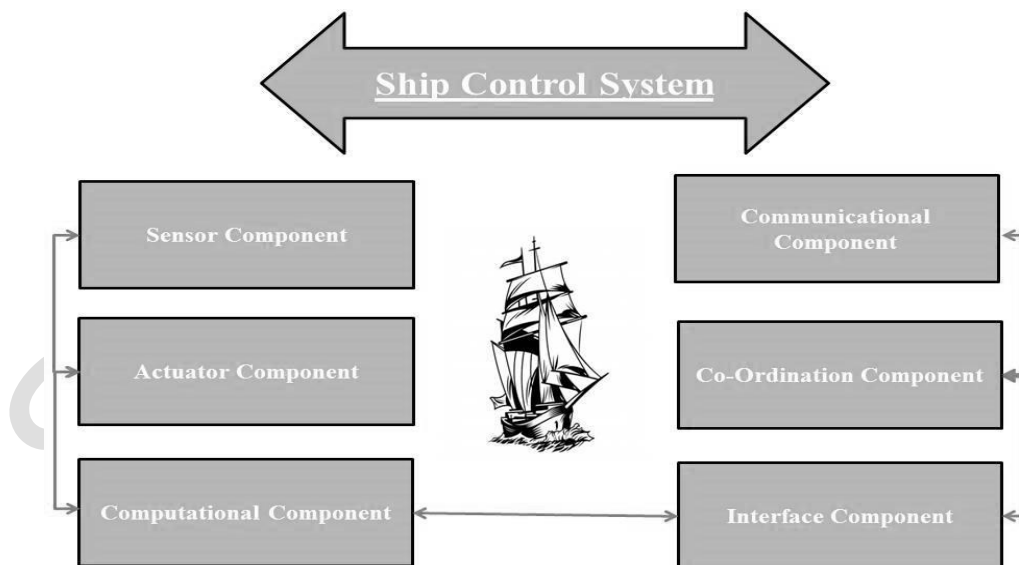
System Architecture

The system requirement and design activity may be modeled as

“a set of components and relationship between these components are represented graphically in order to give the overview of system organization”.

The following diagram represents the architecture of ship control system:

Functional



components of ship control system are,

Sensor Components:

It collects information from system environment about nautical miles covered, tides and waves of the environment.

Actuator Components:

It causes some change in system environment to suit the performance of ship.

Computation Components:

It performs the computation by accepting the input, process the input and gives the required output.

Communication Component:

Establishes communications contact between one ship to another ship in the high sea during emergency time.

Co-ordination component:

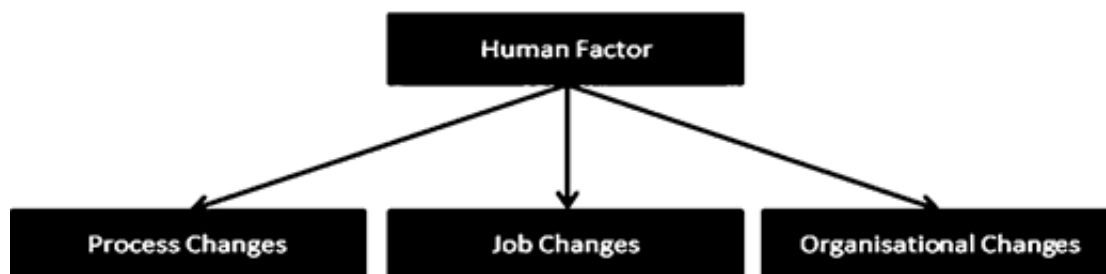
Establishes communications between crew members of the ship to steer the ship in right direction.

Interface component:

It transforms the representation used by one system components into the representation used by other Components.

Human Factors that affects System Environment

All systems have human who use the system in a social and Organizational context. An appropriate user interface should be there for effective System operation. Human factors are important in determining the successful system operation.

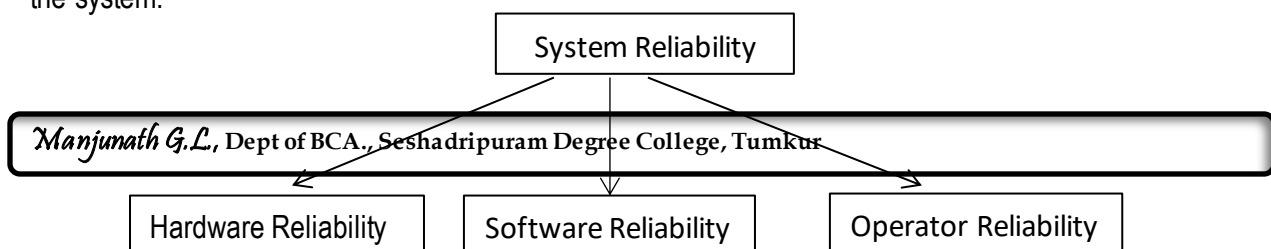


- **Process Changes** : In the organisation if there is a change in process, then the training is required by the worker to cope up with the new system.
- **Job Changes** : As a new and faster system are introduced the worker may have to change the way of work.
- **Organisational Changes** : If the organisation is dependent on the complex system , those who know how to operate the system have a great deal of Political Power.

System Reliability

Reliability is a complex concept. It is consider at the level of system rather than individual components.

The components in the system are dependent and failure in one component can be propagated throughout the system.



- **Hardware Reliability:**
It is the probability of a hardware component failing coordination with the subcomponents.
- **Software Reliability:**
Software failure is usually distinct from hardware failure, how likely that is a software component will produce an incorrect output.
- **Operator Reliability:**
How likely an Operator of a system will make an error.

Requirements and Specification

Software Requirement Specification:

It is a perfect detailed description of the behavior of the System to be developed.

The SRS document is a formal agreement between the developer and the customer covering the functional and Non-functional requirements of the software to be developed. The first step of SRS document is to understand, problem, goals & constraints. The second step involves study about the specification in the first step & validating the requirement specification document.

Why SRS document?

- The use of this document is to capture user requirements & highlight any inconsistency, conflicting & requirements.
- The client does not have software development knowledge so as the developer about client's problem. So that SRS document acts as bridge between both.

Classification of software requirements:

- *Functional Requirements*
- *Non-Functional Requirements*

Functional Requirements

- **Functional requirements for a system describe of what the system should do.** These requirements depend on the type of software being developed.
- Functional Requirement specification of a system should be both complete and consistent.
- **Completeness** means that all services required by the user should be define.
- **Consistency** means the SRS is consistent if, and only if, no subset of individual requirements described in its conflict.

Non functional requirements

- **Nonfunctional requirement is a statement of how a system must behave:** it is a constraint upon the system behavior.

- Nonfunctional requirements specify all the remaining requirements not covered by functional requirements.
- Constraints on the services of function offered by the system such as timing constraints, constraints on development process, standards etc.,
- Nonfunctional requirements are properties, characteristics or quality that a software product must have for it to do a task well.
- Nonfunctional requirements include safety, usability, reliability and performance requirements.

<u>Functional Requirements</u>	<u>Non Functional Requirements</u>
A functional requirement defines a system or its component.	A non-functional requirement defines the quality attribute of a software system.
It specifies "What should the software system do?"	It places constraints on "How should the software system fulfill the functional requirements?"
Functional requirement is specified by User.	Non-functional requirement is specified by technical peoples e.g. Architect, Technical leaders and software developers.
It is mandatory.	It is not mandatory.
It is captured in use case.	It is captured as a quality attribute.
Defined at a component level.	Applied to a system as a whole.
Helps you verify the functionality of the software.	Helps you to verify the performance of the software.
Functional Testing like System, Integration, End to End, API testing, etc are done.	Non-Functional Testing like Performance, Stress, Usability, Security testing, etc are done.
Usually easy to define.	Usually more difficult to define.
Example 1) Authentication of user whenever he/she logs into the system.	Example 1) Emails should be sent with a latency of no greater than 12 hours from such an activity.

2) System shutdown in case of a cyber attack.

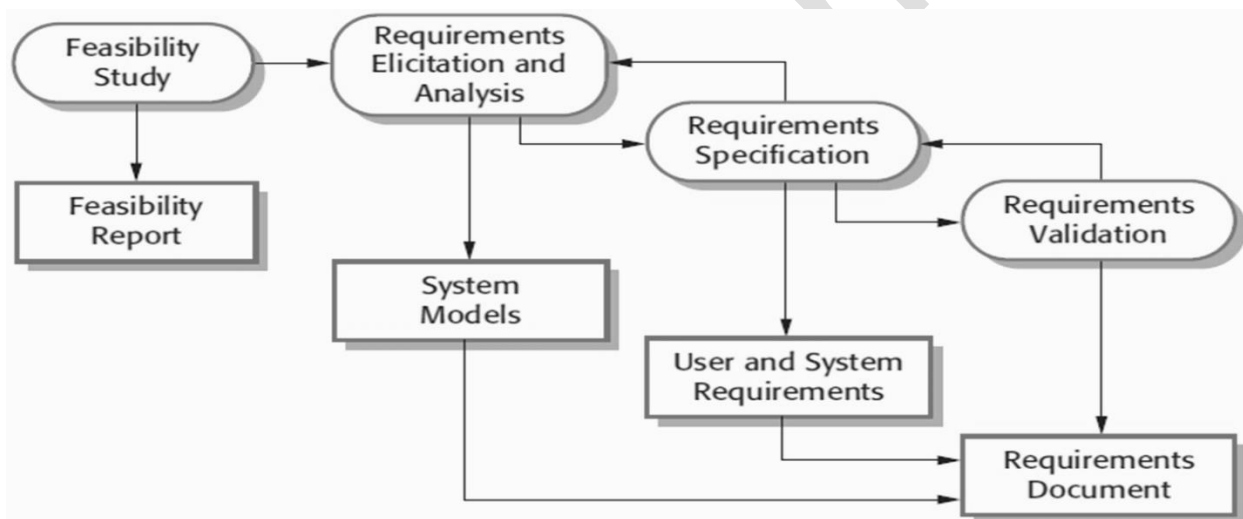
2) The processing of each request should be done within 10 seconds

Requirement Engineering Process

The process of establishing the services that the customer requires from the system and the constraints under which it operates is called Requirement Engineering Process.

The activities involved in the requirement engineering include,

- **Feasibility study**
- **Requirement elicitation and analysis**
- **Requirement specification**
- **Requirement validation**
- **Requirement management**



A. Feasibility Study:

The objective behind the feasibility study is to create the reasons for developing the software that is acceptable to users, flexible to change and confirmable to established standards.

Types of Feasibility:

1. **Technical Feasibility** - Technical feasibility evaluates the current technologies, which are needed to accomplish customer requirements within the time and budget.
2. **Operational Feasibility** - Operational feasibility assesses the range in which the required software performs a series of levels to solve business problems and customer requirements.
3. **Economic Feasibility** - Economic feasibility decides whether the necessary software can generate financial profits for an organization.

B. Requirement Elicitation & Analysis

- This is a process of deriving the system requirements through observation of existing system, discussion with potential user & procedures, task analysis and so on.
- This may involve the development of one or more system models & prototypes.

C. Requirement specification:

- It is the activity of translating the information gathered during the analysis activity into a document that defines set of requirements.
- Abstract statement of the system requirement for the customer & end user of the system.
- System requirements are those detailed description of the functionality to be provided.

D. Requirement Validation:

This activity checks requirements for real-time environment, consistency & completeness.

* *Elicitation means to bring out*

Components of SRS:

- **Functional Requirements.**
 - **Performance Requirements.**
 - **Interface Requirements.**
 - **Operational Requirements.**
 - **Verification Requirements**
 - **Acceptance Testing Requirements**
 - **Documentation Requirements**
 - **Quality Requirements**
 - **Safety Requirement.**
 - **Reusability Requirements**
 - **Reliability and Maintainability.**
-
- **Functional Requirements :** This is a subset of overall system requirements. This includes h/w & s/w & also describes how the system operates under normal condition & response to s/w failure or invalid input to the system.
 - **Performance Requirements:** This can be measured in value i.e., rate, frequency, speed & travel
 - **Interface Requirements:** This is separated for h/w & s/w. This requirement specify any interface standard is requested
 - **Operational Requirements:** The field views of the system are notified in this stage.
How system will operate & communicate?
What are the operator syntax or notations?
How error messages should be displayed?
 - **Verification Requirements:** This specifies how customer accepts after completion of the project.
This also specifies how functional & performance requirements to be measured.

- **Acceptance Testing Requirements:** This provides details of the tests to be performed for customer acceptance in document.
- **Documentation Requirements:** This specifies what documents are to be supplied to the clients either through the project or at the end of the project
- **Quality Requirements:** It specifies whether product should meet international or local standards.
- The quality factors are correctness, reliability, efficiency, integrity, usability, maintainability, flexibility, portability & reusability.
- **Safety Requirement.:** This specifies safety steps to be taken for protection of human, equipment & data i.e., protecting from moving parts electrical circuiting & other physical dangers.
- **Reusability Requirements and Maintainability:** This states that s/w must perform function understated condition for a given period of time.
- This maintenance of the sight is used for h/w & s/w changes in the system.

Characteristics of a good SRS document

1. Completeness:

The SRS should include all types of requirements to be specified.

2. Consistent:

Requirements in SRS are said to be consistent if there are no conflicts between any set of requirements.

Examples of conflict include differences in terminologies used at separate places,

3. Traceable:

Tracing the reference which help in modification have made to requirement to bring out its current state.

4. Unambiguous:

This means not having two or more possible meanings. Each requirement can have only one interpretation.

5. Verifiable:

The s/w requirement specification must be verifiable that it contains all of the requirements from the user.

Benefits of a good SRS document

1. Establish basis for agreement between client the supplier.
2. Reduces development cost.
3. Helps in removal of inconsistencies & misunderstanding.
4. Helps in validation of final product.

Requirement Elicitation & Analysis

It is a process of gathering the information or requirements for the system from stake holders.

Requirement elicitation and analysis is concerned with the process of analyzing requirement to,

- Detect and resolve conflicts between requirements
- Discover the bounds of the software and how it must interact with its environment

Techniques for Requirement Elicitation & Analysis,

1. View point orientation elicitation
2. Scenarios
3. Ethnography

1. View point orientation elicitation:

View point orientation elicitation approach takes different views given by different stake holders.

View points can be used as a way of classifying stake holder & other source of requirements.

There are three generic types of view points,

- a. **Interactor viewpoints:** Represents people who directly interact with the system.

Ex: In bank customers & bank account database.

- b. **Indirect viewpoints:** Represents people who do not use the system but who influence the requirements

Ex: In Bank indirect view points are management of the bank & the bank security staff.

- c. **Domain viewpoints:** Represents characteristics and constraints which influence the System requirements.

Ex: Consider an ATM as example where present bank customer, h/w maintenance, customer security staff of the bank can access ATM

2. Scenarios:

- These are real life examples how a system can be used.
- They are helpful in requirements elicitation as people can relate to scenarios more rapidly than abstract statements of what they require from system.
- A scenario should include a description of,
 - The Starting situation
 - A description of the normal flow of events
 - A description of what can go wrong and how it can be handled
 - Information about other concurrent activities
 - A description of the system state

3. Ethnography:

Ethnography is an observational technique that can be used to understand social and organizational requirements.

Ethnography is particularly effective at discovering two types of requirements;

- a. Requirements that are derived from the way in which people actually work rather than the way in which definitions say correctness of the word.
- b. Requirements that are derived from co-operation & awareness of other people activities.

The Ethnography informs the development of the prototype so that fewer prototype refinement cycles are required.

Prototyping focuses the Ethnography by identifying problem & questions that can be discussed with Ethnographer.

* *Ethnographer is one who observes social & organizational requirement factors.*

Different Specification Method.

1. **Structured natural language:**

Defining standard form or template to express the requirement specification using natural language.

2. **Design description language.**

This method concentrates on operation & constraints of the system through programming language or design diagram language.

3. **Requirement specification language**

Various special purpose language have been designed to express software requirement such as PSL(Problem Statement Language), PSA (Problem Statement Analyzer)

4. **Graphical notation:**

Structured analysis & design technique (SADT) has a many complex graphical vocabulary & is mostly used by specialists.

5. **Mathematical specification**

These notation based on formal mathematical concepts such as sets etc.,

SYSTEM MODELS

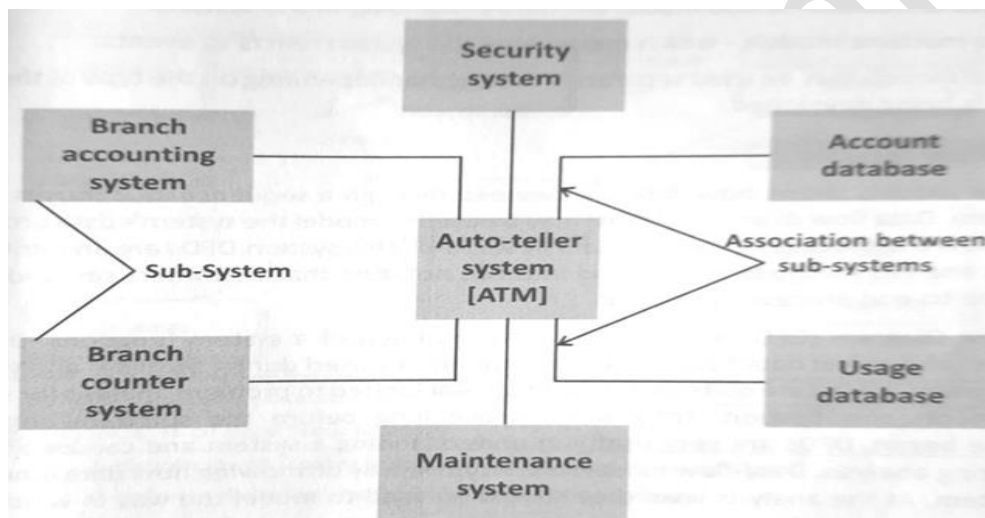
System models are graphical representation that describes, business processes, the problem to be solved and the system that is to be developed.

System Models are classified based on different perspective,

1. **Context Model**
2. **Behavioral Model**
3. **Data Models**
4. **Object Modeling**
1. **Context Model**

It is an architectural model that describes environment and boundary of the system.

- This shows the process activities and how data transfers between one system to another.
- They show what lies outside the system boundaries and they do not show relationships.
- High level architecture models are usually expressed as the simple block diagrams where each sub-system is replaced by named rectangle and lines indicates association between sub-system.



Context model of an ATM

- Security system support machine maintenance.
- Usage database monitors how the network used.
- Branch counter system provides services such as backup and printing.
- Branch accounting system maintains account transactions.
- Account database manages account transaction data
- Maintenance system manages the ATM system maintenance including electricity supply etc.,

2. Behavioral Model

This model describes the overall behavior of the system.

- It describes the internal processing of a system.
- It specifies how one process is related with another within the system boundary.

Types :

- a). Data Flow Model b). State machine model.

a) Data Flow model :


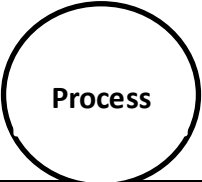
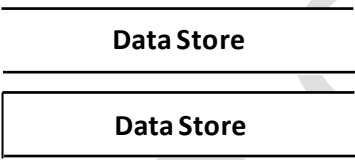
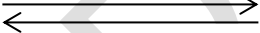
This model shows how the data is processed through a sequence as it moves through the system. It is a graphical representation of data flow and functionality of the system. It is capable of representing the incoming flow, outgoing flow of data as well as storage of data. They are also known as **Data Flow Diagram(DFD)**

Types :**i) Logical DFD****ii) Physical DFD**

i) **Logical DFD** : It represents the system processes and dataflow within the system.

ii) **Physical DFD**: It represents the actual data flow implemented in the system.

DFD Components:

	Entities represents the source(input) & destination(output) of data, Entities are denoted by Rectangle
	Process represents the task performed on data. Processes are denoted by circle and also rounded rectangle.
	Data Storage or database table represents the data to be stored in the system. It is denoted by the two parallel lines or a rectangle with one smaller side missing.
	Movement of data between source and destination are represented by flow lines

DFD Rules :

- Each process should have at least one input and one output.
- Each data store should have at least one data flow-in and one data flow-out.
- Data stored in a system must go through a process.
- All processes in a DFD goto another process or a data store.

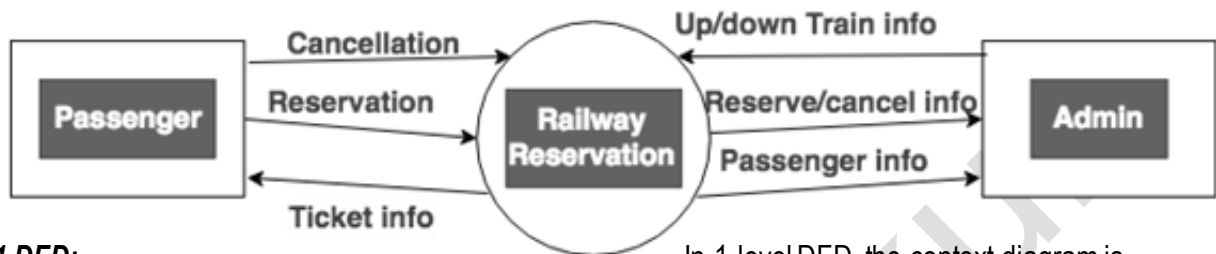
Levels of DFD :

- ✓ 0 level DFD
- ✓ 1st level DFD
- ✓ 2nd Level DFD

✓ 3rd Level DFD so on.....

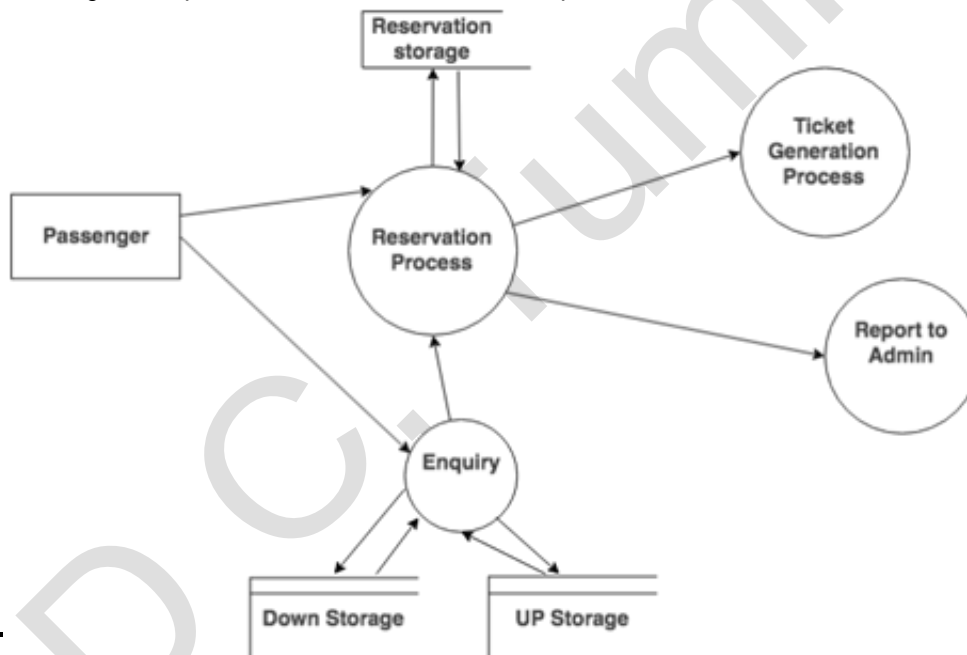
Level 0 is the highest abstraction level DFD.

Level-0 declares the entire information system as one diagram considering all the underlying details. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.



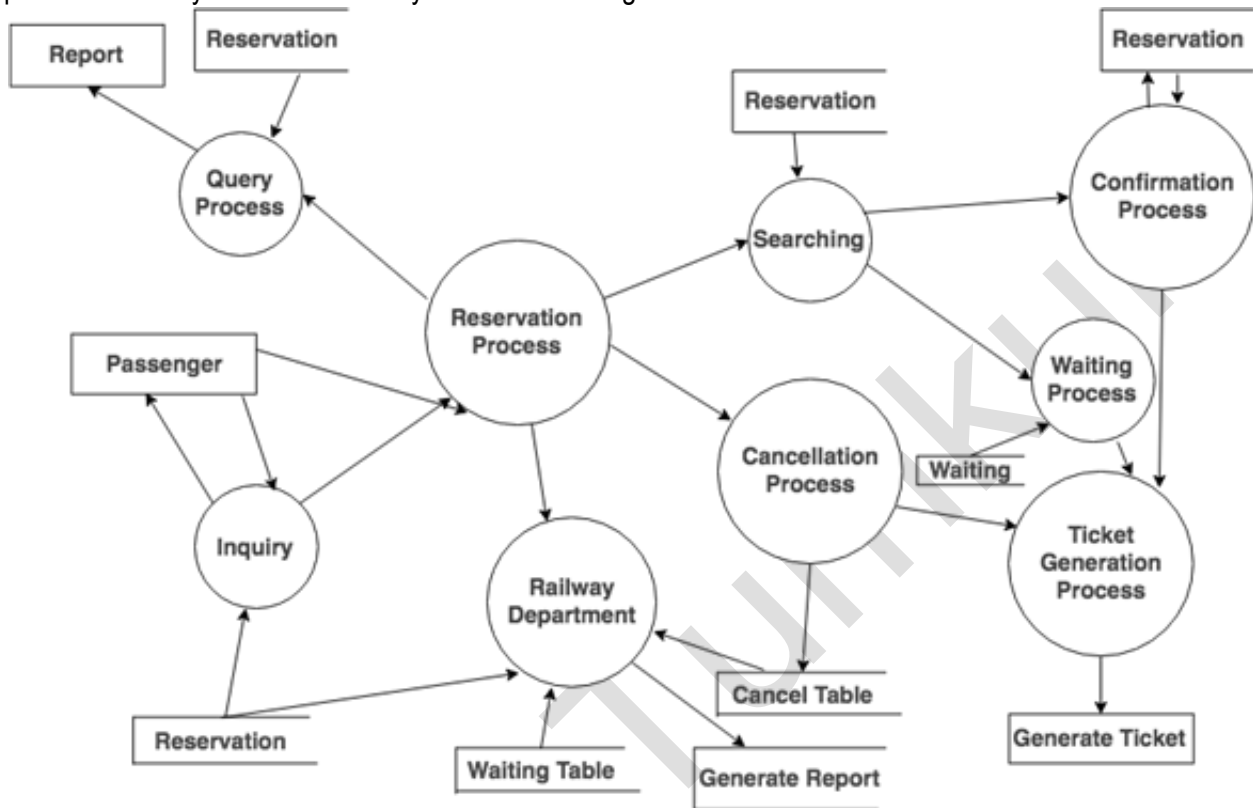
Level-1 DFD:

In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into sub-processes.



Level-2 DFD:

Level-2 DFD goes one step deeper into parts of level-1 DFD. It can be used to plan or record the specific/necessary detail about the system's functioning.



Advantages of DFD:

- A simple graphical technique which is easy to understand
- It helps to define the boundaries of the system
- It is used as a part of system documentation file
- It explains the logic behind the data flow within the system

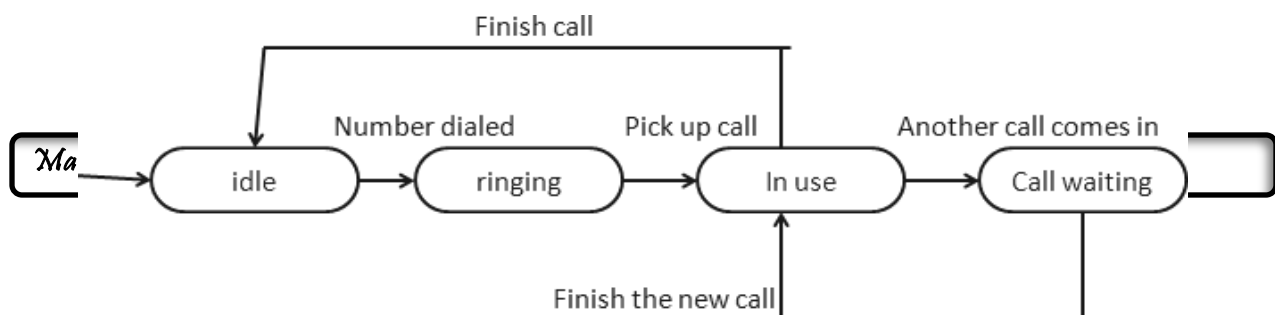
Disadvantages of DFD:

- DFD undergoes a lot of alteration before going to users, so makes the process little slow
- There is no individual standard for producing different format of DFD

b). State Machine Model / Event Model:

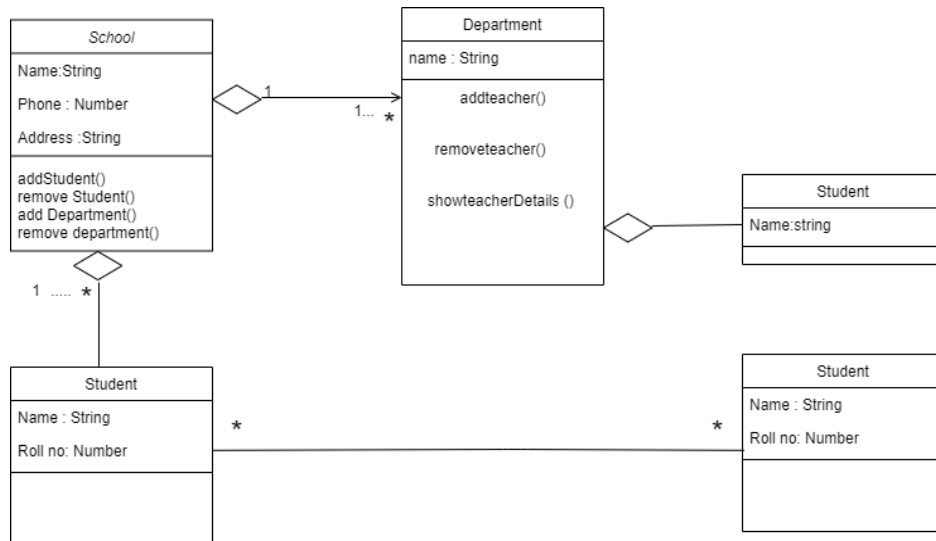
A state machine model describes how a system response to internal or external events.

- This model shows system state and event that cause transition from one state to another state
- It does not show the flow of data within the system.
- According to event it continue for further process it don't wait for completion of previous event



3. Data Model / Semantic Model:

Semantic data model describe the logical structure of the data which is imported and exported by the system. Models which represent processing of logical data belonging to this system and such models are often referred as semantic data models. The entities attributes and relationship existing between the entities are the main aspect of these models

**Advantages:**

- It can be applied as a major tool for name management.
- It can be used for storing large information at organization level

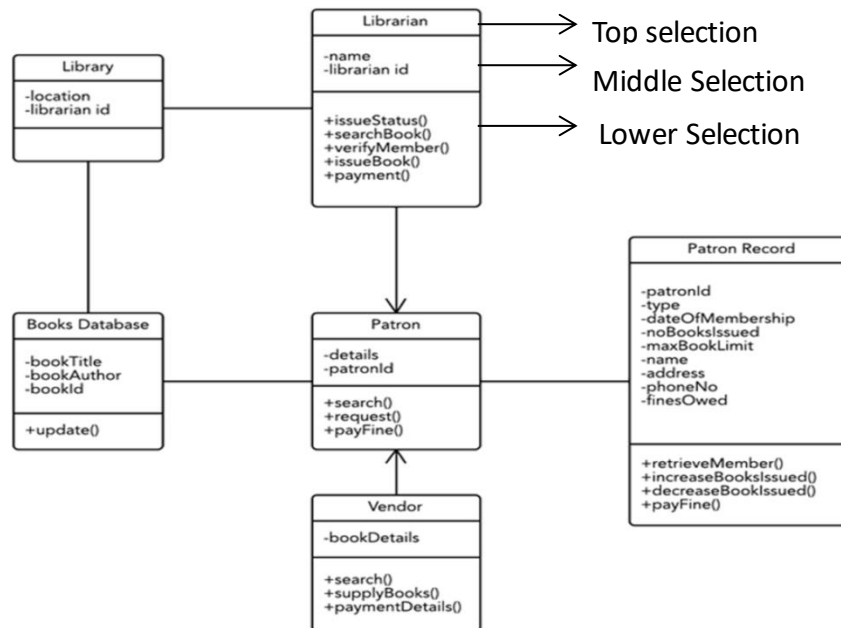
4. Object Models

Data models combine some of the uses of data flow and semantic data models.

It is also helpful for showing how entities in the system may be classified and composed of other entities. It is used to represent both system data and its processing. Developing object models during requirements analysis is usually simplifies that transition to object oriented design and programming. Standard is maintained for object modelling is also called as **Unified Modelling Language(UML)**

An object class in UML has three sections:

- Top Section: Name of the object class
- Middle Section: Class attributes.
- Lower Section: Operations associated with object class.



Important Questions:

1 Mark:

1. What are Bespoke products?
2. Give any two examples for functional requirements.
3. Give any two examples for non-functional requirements.
4. Define Ethnography.
5. Define context model.
6. What is Validation & Testing?
7. Define software Engineering
8. What are functional requirements?
9. Define prototype.
10. What are Generic products?
11. Define Evolutionary prototyping.

3 Marks:

12. Explain SDLC with neat diagram.
13. Explain human factor that affects the system and its environment.
14. State & Explain context model with neat diagram.
15. Explain data flow model with an example.
16. Explain COTS system procurement method.

5 Mark:

17. What is professional & ethical responsibilities of a software engineer.
18. Define behavioral model. Explain state mission mode.
19. Explain waterfall model with neat diagram.
20. Explain view point oriented elicitation technique with neat diagram.
21. Explain detail structure of SRS document.
22. ***Write the difference between Evolutionary & throw away prototyping.
23. Explain requirement elicitation & analysis process.

7 Marks:

24. Define feasibility. Explain waterfall model with neat diagram. Mention its advantages and disadvantages.
25. Explain system engineering process with neat diagram.
26. Explain different techniques used for requirement elicitation and analysis.
27. What is DFD? Explain with an example.
28. What is system procurement? Explain different system procurement methods.
29. Define object model. Explain its types with neat diagram.

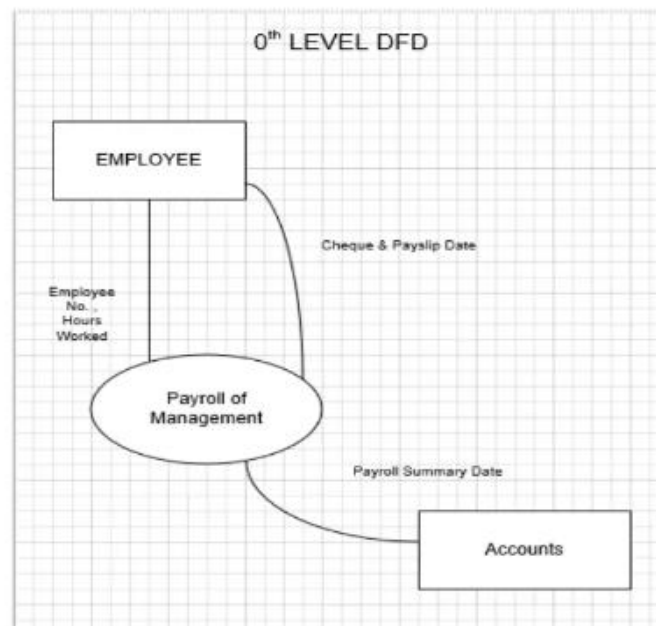


Fig 1.2. DFD (level 0)
(Payroll Management System)

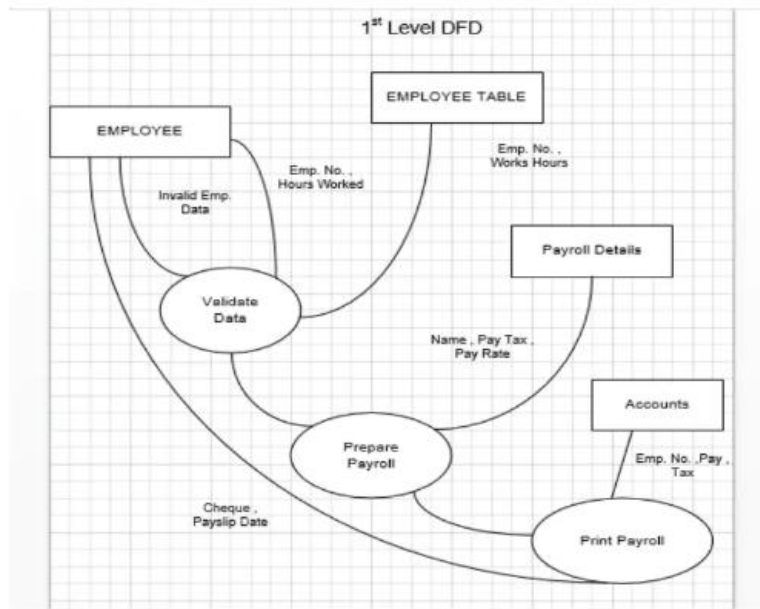


Fig 1.3. DFD (level 1)
(Payroll Management System)

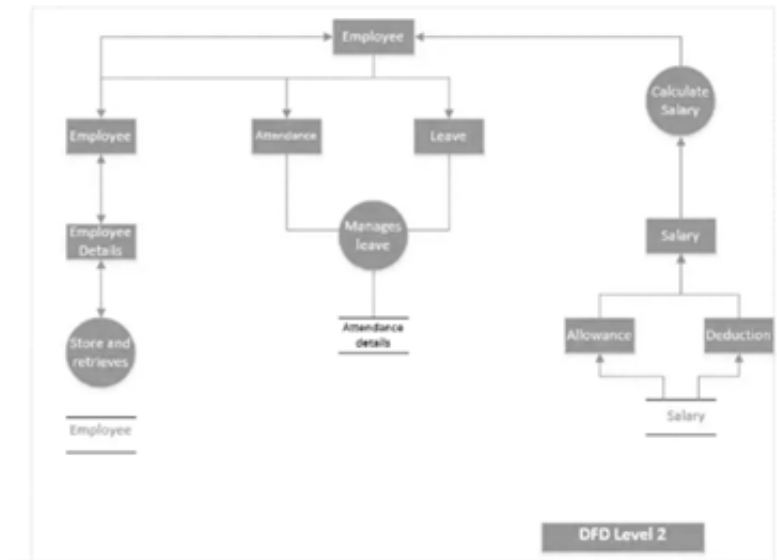


Fig 1.4 DFD (level 2)
(Payroll Management System)